

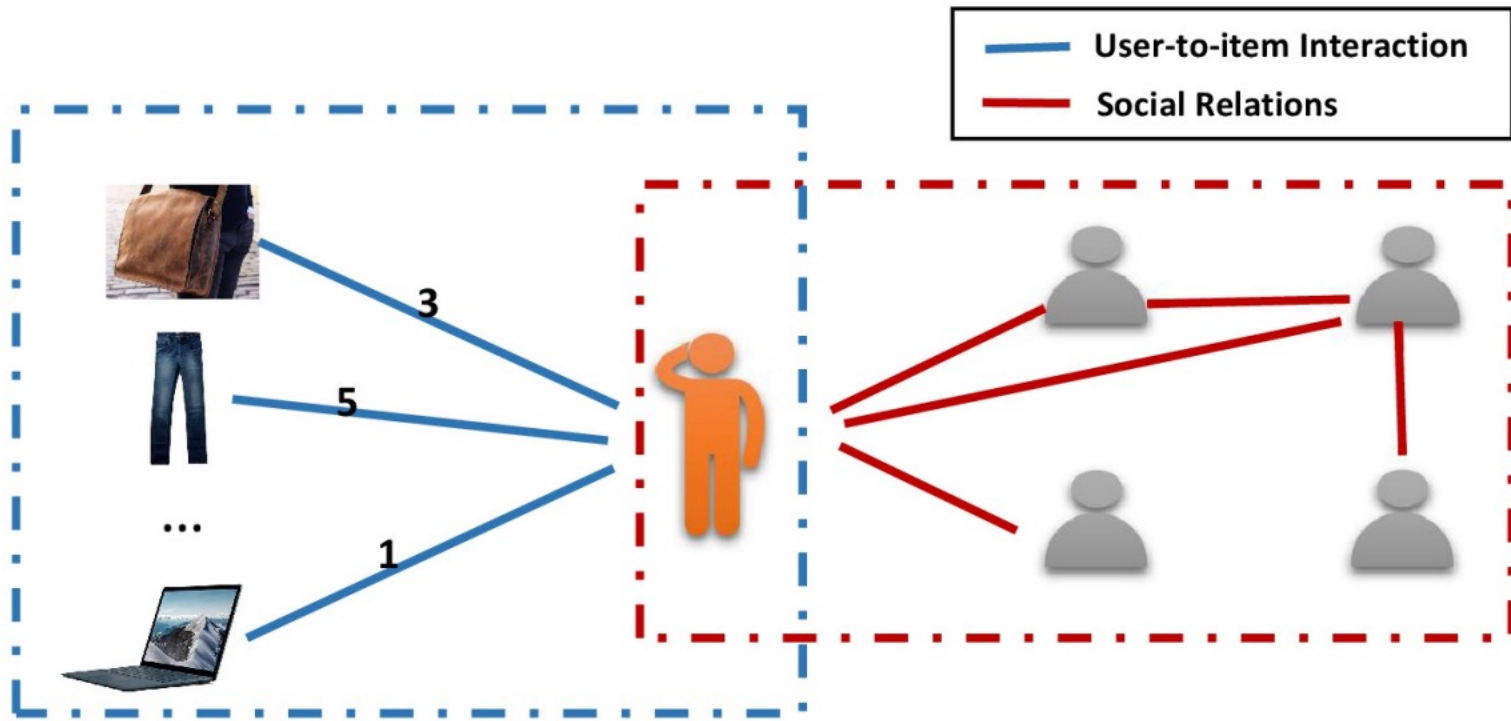


ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG



Mạng nơ-ron đồ thị

Nguyễn Phi Lê



Mục lục

- Nhắc lại các kiến thức cơ bản về đồ thị
- Graph embedding
- Graph neural network

Các kiến thức cơ bản về đồ thị

Tại sao cần học trên đồ thị?

- Graph: cấu trúc dữ liệu cho phép biểu diễn nhiều loại dữ liệu phức tạp
 - Ví dụ: biểu diễn thông tin mạng xã hội, mạng giao thông, tương tác giữa các protein, mạng tri thức, ...
- Graph-based tasks được ứng dụng giải quyết nhiều bài toán thực tế
 - Node classification
 - Phát hiện các sự liên quan giữa gen và bệnh
 - Đề xuất thuốc cho bệnh nhân
 - Phát hiện bất thường trong mạng
 - Link prediction
 - Xây dựng hệ gợi ý
 - Dự đoán hiệu ứng phụ của thuốc
 - Phân tích hiệu quả của thuốc
 - Xây dựng đồ thị tri thức

Sử dụng dữ liệu dạng đồ thị

- Học máy truyền thống
 - Các mẫu dữ liệu được coi như các thực thể độc lập
- Graph
 - Các nút mạng được kết nối với nhau, có quan hệ mật thiết với nhau



- Giải pháp 1: xây dựng 1 cơ chế học máy riêng để xử lý dữ liệu đầu vào dạng đồ thị
 - Xem xét thông tin của bản thân các nút mạng và các hàng xóm của chúng
- Giải pháp 2: chuyển đổi thông tin của các nút mạng
 - Làm phẳng đồ thị
 - Xây dựng ánh xạ từ đồ thị sang tập các vector biểu diễn thông tin các nút mạng

Nhắc lại kiến thức về đồ thị

- Định nghĩa
 - $G(V, E)$, V : tập các đỉnh, E : tập các cạnh
- Một số khái niệm hay dùng
 - Node degree (bậc của đỉnh) = tổng số các đỉnh có kết nối trực tiếp tới đỉnh hiện tại
 - graph diameter (đường kính đồ thị) = độ dài đường đi ngắn nhất giữa 2 đỉnh
 - adjacency matrix (ma trận kề)
 - k-hop neighbors: các đỉnh có đường đi ngắn nhất tới đỉnh hiện tại không vượt quá k
 - Subgraph (đồ thị con): đồ thị có tập đỉnh và tập cạnh là tập con của đồ thị hiện tại
 - Walk (đường đi): một chuỗi tuần tự của các đỉnh-cạnh; bắt đầu từ 1 đỉnh và kết thúc tại 1 đỉnh
 - Trail: 1 đường đi có các cạnh không lặp lại
 - Path: 1 đường đi có các đỉnh không lặp lại

Các ký hiệu

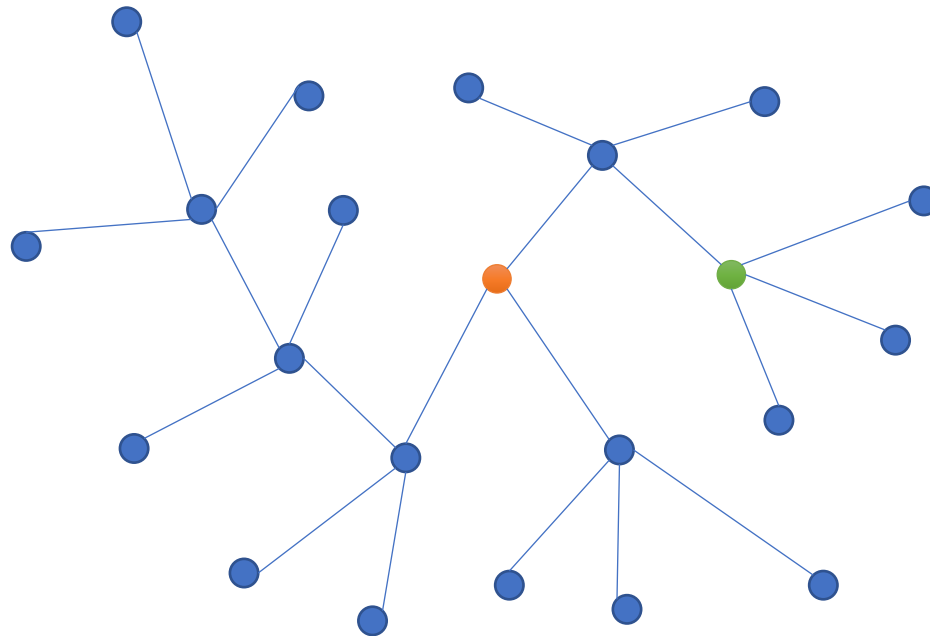
- $G(V, E)$: đồ thị
- A : ma trận kề
- V : tập các đỉnh, v_i : đỉnh thứ i
- E : tập các cạnh, e_{ij} : cạnh nối v_i và v_j
- $N(v_i)$: tập các hàng xóm của v_i
- $d(v_i)$: bậc của v_i
- $dis(v_i, v_j)$: khoảng cách giữa v_i và v_j = độ dài đường đi ngắn nhất từ v_i tới v_j

Độ trọng tâm của một đỉnh

- Độ trọng tâm (Centrality) của 1 đỉnh:
 - Thể hiện mức độ trung tâm, sự quan trọng của một đỉnh trong đồ thị
 - Thông thường, những đỉnh có nhiều cạnh kết nối tới được coi là có độ trọng tâm cao
 - Một số định nghĩa thường được dùng để đo độ trọng tâm: Degree Centrality, Eigenvector Centrality, Katz Centrality, Betweenness Centrality
- Degree Centrality: định nghĩa theo bậc của đỉnh, đỉnh có bậc càng cao thì độ trọng tâm càng lớn
 - Coi tất cả các hàng xóm như nhau, chỉ quan tâm tới số lượng hàng xóm, không quan tâm tới độ quan trọng của hàng xóm
- Eigenvector Centrality: định nghĩa theo độ quan trọng của các hàng xóm
 - Đỉnh có càng nhiều hàng xóm quan trọng thì càng quan trọng

Độ trọng tâm của một đỉnh

- Eigenvector Centrality: định nghĩa theo độ quan trọng của các hàng xóm
 - Đỉnh có càng nhiều hàng xóm quan trọng thì càng quan trọng



Độ trọng tâm của một đỉnh (tiếp)

- Eigenvector Centrality

Độ trọng tâm của đỉnh v_i

$$c_e(v_i) = \frac{1}{\lambda} \sum_{j=1}^N A_{i,j} \cdot c_e(v_j),$$

$A_{ij} = 1$ nếu v_j kề v_i

Tổng độ trọng tâm của tất cả hàng xóm

$$\rightarrow \mathbf{c}_e = \frac{1}{\lambda} \mathbf{A} \cdot \mathbf{c}_e,$$

Ma trận kề

Vector biểu diễn độ trọng tâm của toàn bộ các đỉnh

$$\rightarrow \lambda \cdot \mathbf{c}_e = \mathbf{A} \cdot \mathbf{c}_e.$$

$\rightarrow c_e$ là vector riêng của A tương ứng với trị riêng λ

Độ trọng tâm của một đỉnh (tiếp)

- Eigenvector Centrality:

$$\lambda \cdot \mathbf{c}_e = \mathbf{A} \cdot \mathbf{c}_e.$$

- \mathbf{c}_e là một vector riêng của A
 - \mathbf{c}_e cần có các thành phần dương
 - Định lý: mỗi ma trận vuông với các thành phần đều dương có duy nhất 1 **trị riêng lớn nhất** và **vector riêng tương ứng** có tất cả các thành phần đều **dương**
- chọn **λ là trị riêng lớn nhất của A**

Độ trọng tâm của một đỉnh (tiếp)

- Betweenness Centrality: định nghĩa dựa trên số lượng đường đi đi qua các đỉnh
 - Đỉnh càng nhiều đường đi đi qua thì được coi là càng có độ trọng tâm cao

$$c_b(v_i) = \sum_{v_s \neq v_i \neq v_t} \frac{\sigma_{st}(v_i)}{\sigma_{st}},$$

Betweenness centrality của đỉnh v_i

Tổng số các đường đi ngắn nhất từ s tới t

số các đường đi ngắn nhất từ s tới t và đi qua v_i

Chuẩn hoá bằng cách chia cho tổng số các đường đi ngắn nhất $c_{nb}(v_i) = \frac{2 \sum_{v_s \neq v_i \neq v_t} \frac{\sigma_{st}(v_i)}{\sigma_{st}}}{(N-1)(N-2)},$

Graph embedding

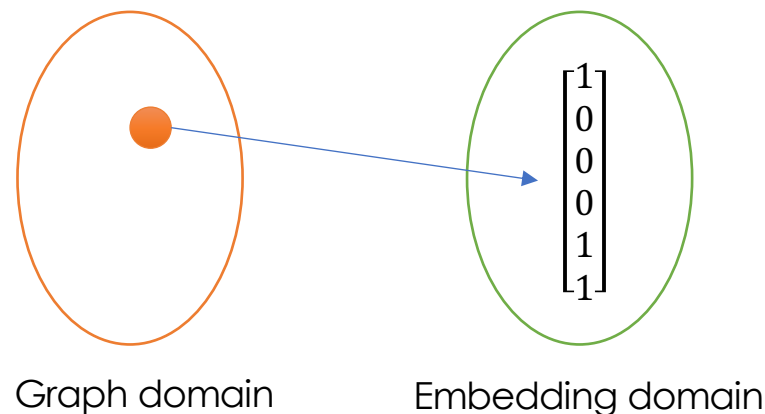
Graph embedding

- Mục tiêu:

- Ánh xạ **mỗi đỉnh** trong một đồ thị cho trước sang **một vector** có số chiều thấp
- **Thông tin** trong miền đồ thị ban đầu (graph domain) **được bảo toàn** nguyên vẹn trong không gian vector mới (embedding domain)

- Câu hỏi

1. Thông tin cần bảo toàn **là gì?**
2. **Làm thế nào** để bảo toàn được thông tin đó?

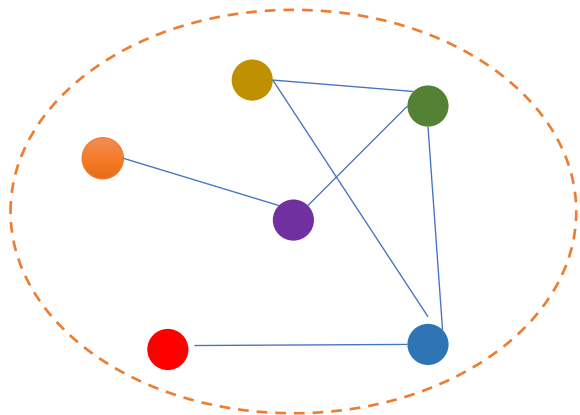


Graph embedding

- Thông tin cần bảo toàn là gì?
 - Thông tin về lân cận của các đỉnh (**neighborhood information**)
 - Vai trò của các đỉnh trong cấu trúc của đồ thị (**structural role**)
 - Độ quan trọng của các đỉnh (node **centrality**)
 - Tính chất co cụm của đồ thị (**community information**)
 - **Làm thế nào** để bảo toàn được thông tin đó?
 - **Tái tạo lại các thông tin cần bảo toàn** trong miền đồ thị, dựa trên các thông tin biểu diễn các đỉnh trong miền vector
 - Một ánh xạ tốt nếu ta có thể tái tạo lại thông tin cần bảo toàn
- Quá trình học máy là quá trình tìm ra ánh xạ **tối thiểu hoá được sai số của thông tin được tái tạo** từ miền vector, so với **thông tin trích xuất từ đồ thị ban đầu**

Graph embedding

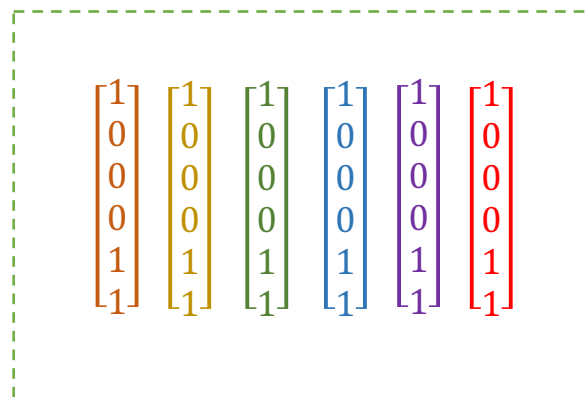
I : Thông tin chúng ta muốn bảo toàn, được trích xuất từ đồ thị ban đầu



Graph domain

Tối thiểu hoá khoảng cách giữa I' và I

I' : Thông tin muốn bảo toàn, được tái tạo từ miền vector



Embedding domain

Graph embedding framework

- **Hàm ánh xạ (Mapping function):** ánh xạ từ miền đồ thị sang miền vector
 - Biểu diễn thông tin của 1 đỉnh dưới dạng 1 vector
- **Bộ trích xuất thông tin (Information extractor):** trích xuất những thông tin chúng ta muốn bảo toàn (I) từ đồ thị ban đầu
- **Bộ tái tạo thông tin (Reconstructor):** Tái tạo lại thông tin chúng ta muốn bảo toàn (\tilde{I}) từ thông tin trong miền vector
- **Mục tiêu (Objective):** Tối thiểu hoá sai số giữa \tilde{I} và I

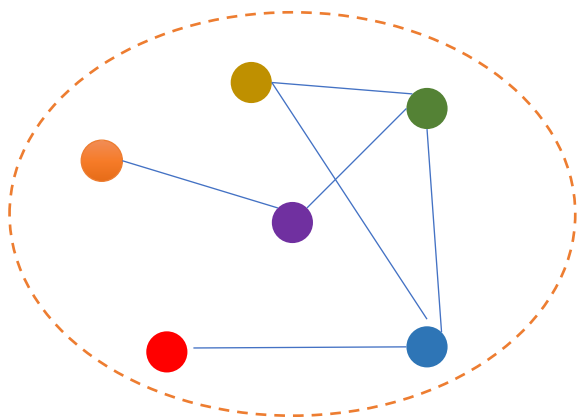
Graph embedding framework

I : Thông tin chúng ta muốn bảo toàn, được trích xuất từ đồ thị ban đầu

$I \stackrel{?}{=} I'$
Objective

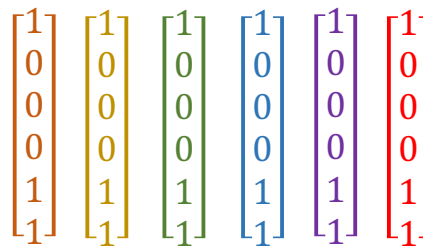
I' : Thông tin muốn bảo toàn, được tái tạo từ miền vector

Information extractor



Graph domain

Mapping function



Embedding domain



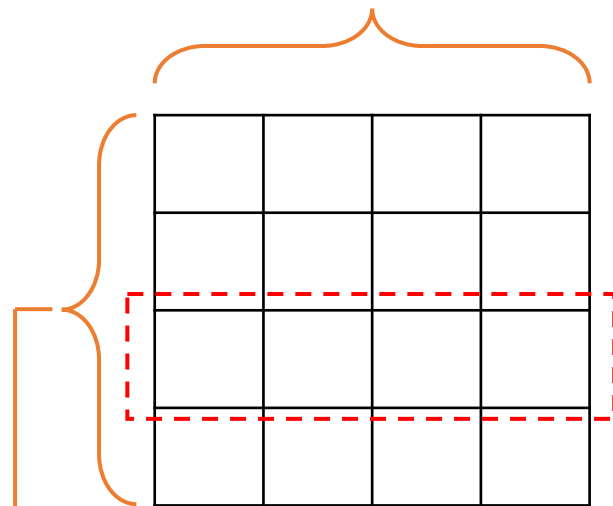
Reconstructor

Graph embedding framework

- Mapping function
 - Phương pháp đơn giản nhất: table lookup
- Information extractor
 - Phương pháp thường dùng nhất: Random walk-based
- Objectives
 - Bảo vệ tính xuất hiện cùng nhau (**Co-occurrence**) của các đỉnh
 - Bảo vệ vai trò của các đỉnh trong cấu trúc của đồ thị (**Structural role**)
 - Bảo vệ trạng thái của các đỉnh (**Node status**)

Table lookup

d : số chiều của vector trong miền vector



$W^{N \times d}$

Mã trận tham số mà chúng ta cần học

u_i : embedding vector của v_i

$$f(v_i) = e_i^T \cdot W$$

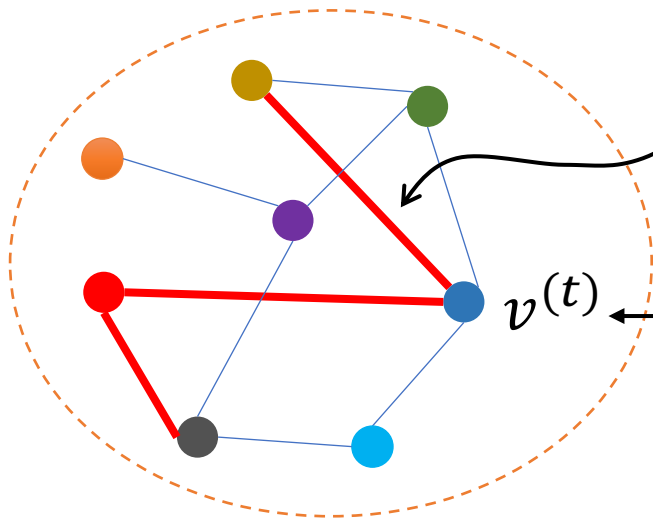
e_i : One-hot vector; $e_i[i] = 1$ and $e_i[j] = 0$

N : số lượng đỉnh của đồ thị

Các thuật toán Walk-based

I: Thông tin chúng ta muốn bảo toàn,
được trích xuất từ đồ thị ban đầu

Information extractor



Graph domain

Một walk với độ dài 3

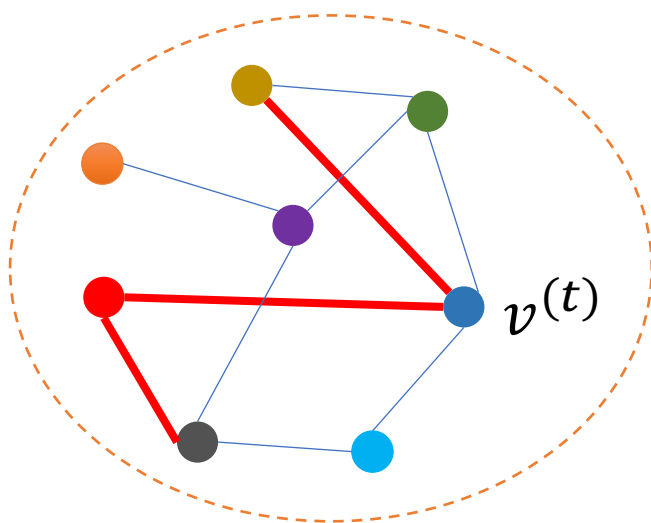
Khi đang đứng tại đỉnh $v^{(t)}$ → làm sao để quyết định đỉnh tiếp theo là đỉnh nào?

→ thường định nghĩa hàm xác suất đi tới đỉnh $v^{(t+1)}$ từ đỉnh $v^{(t)}$?

Random walk

I : Thông tin chúng ta muốn bảo toàn, được trích xuất từ đồ thị ban đầu

Information extractor



Graph domain

- Thuật toán random
 - Bắt đầu từ đỉnh $v^{(0)}$, đi ngẫu nhiên tới 1 đỉnh lân cận của nó
 - Lặp lại quá trình này cho tới khi đi qua T đỉnh \rightarrow thu được 1 walk với độ dài T

I : là tập của các walks

Bảo toàn node co-occurrence

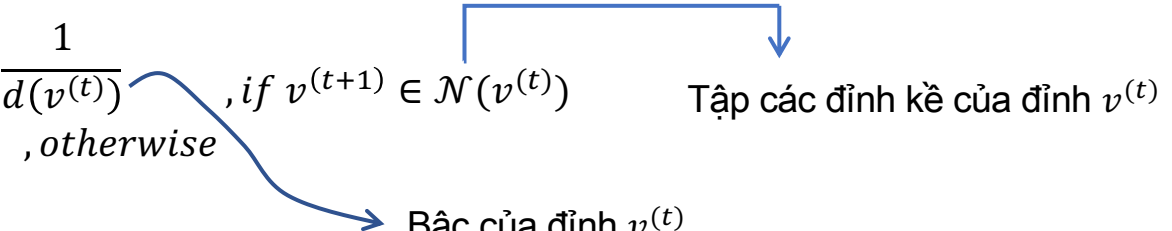
- Thuật toán điển hình: DeepWalk (Perozzi et al., 2014)

- Information extractor:

- Thực hiện random walk để tạo ra tập các walks
- Mỗi walk có thể coi là 1 câu
- Tập các đỉnh có thể coi là tập các từ
- DeepWalk
 - Mỗi đỉnh sẽ là điểm bắt đầu của đúng γ walks
 - Mỗi walk có độ dài T

Các đỉnh cùng xuất hiện trong một walk được coi là có co-occurrence cao

$$p(v^{(t+1)}|v^{(t)}) = \begin{cases} \frac{1}{d(v^{(t)})}, & \text{if } v^{(t+1)} \in \mathcal{N}(v^{(t)}) \\ 0, & \text{otherwise} \end{cases}$$



Bậc của đỉnh $v^{(t)}$

Tập các đỉnh kề của đỉnh $v^{(t)}$

Bảo toàn node co-occurrence: DeepWalk

- Information extractor:

- R : tập các walks; $W \in R$: một walk
- Với mỗi $W \in R$
 - Duyệt toàn bộ các đỉnh $v^{(i)}$: thêm $(v^{(i-j)}, v^{(i)})$ và $(v^{(i+j)}, v^{(i)})$ vào I nếu $j \leq w$; w là một tham số quy định ngưỡng
 - Co-occurrence của 2 đỉnh được ký hiệu là (v_{con}, v_{cent})
→ Tập thông tin cần bảo toàn là $I = \{(v_{con}, v_{cen})\}$
 - v_{cen} : là đỉnh ta đang xét, v_{con} : là 1 ngữ cảnh của v_{cen}

Bảo toàn node co-occurrence: DeepWalk

- **Reconstructor**

- Dự đoán xác suất sẽ quan sát được các cặp đỉnh thuộc I dựa trên các thông tin trong miền vector
 - Biết biểu diễn trong miền vector của 2 đỉnh v_{con}, v_{cen} tương ứng là u_{con}, u_{cen}
 - Dự đoán xác suất v_{con}, v_{cen} sẽ xuất hiện cùng nhau

$$p(v_{con}|v_{cen}) = \frac{\exp(u_{con}^T \times u_{cen})}{\sum_{v \in V} \exp(u^T \times u_{cen})}$$

Phản ánh độ tương đồng của u_{cen} và u_{con}
Độ tương đồng giữa u_{cen} và u_{con} càng cao
Thì xác suất này càng cao

Tất cả các đỉnh trong đồ thị

u là biểu diễn của v

Bảo toàn node co-occurrence: DeepWalk

- Ý nghĩa của công thức

- $p(v_{con}|v_{cen}) = \frac{\exp(u_{con}^T \times u_{cen})}{\sum_{v \in V} \exp(u^T \times u_{cen})}$



Là xác suất khi ta nhìn thấy v_{cen} thì đồng thời cũng nhìn thấy v_{con} gần đấy (i.e., v_{con} là đỉnh lân cận của v_{cen})

Chúng ta có: V là tập tất cả các đỉnh của đồ thị $\rightarrow V =$ tập tất cả các khả năng mà ta có thể nhìn thấy

\rightarrow Có thể coi như 1 bài toán phân loại với tập nhãn là V

v_{cen} có vector đặc trưng là $u_{cen} \rightarrow p(v_{con}|v_{cen})$ là xác suất để v_{cen} có nhãn là v_{con}

\rightarrow Sử dụng hàm softmax

$\rightarrow p(v_{con}|v_{cen}) = \frac{\exp(u_{con}^T \times u_{cen})}{\sum_{v \in V} \exp(u^T \times u_{cen})}$ \longrightarrow Tổng trên tất cả các nhãn

Bảo toàn node co-occurrence: DeepWalk

- Objective

- Xác xuất khôi phục lại thông tin I được mô hình hoá bởi:

$$I' = Rec(I) = \prod_{(v_{con}, v_{cen}) \in I} p(v_{con} | v_{cen}) = \prod_{(v_{con}, v_{cen}) \in Set(I)} p(v_{con} | v_{cen})^{\#(v_{con}, v_{cen})}$$

$$\mathcal{L}(\mathbf{W}_{con}, \mathbf{W}_{cen}) = -\log I' = - \sum_{(v_{con}, v_{cen}) \in Set(I)} \#(v_{con}, v_{cen}) \cdot \log p(v_{con} | v_{cen})$$

Câu hỏi: Tại sao hàm loss lại là log của I' mà không phải là chính I' ?

Đọc thêm:

- Hierarchical Softmax
- Negative Sampling (Mikolov et al., 2013)

Một số thuật toán tương tự DeepWalk

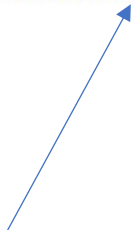
- node2vec (Grover and Leskovec, 2016)
 - Dựa trên **biased-random walk**
 - Nhắc lại về **deepwalk**: chọn đỉnh $v^{(t+1)}$ theo phân phối đều trên tập các đỉnh kề của v^t
 - **biased-random walk**: xem xét cả v^t và v^{t-1} khi lựa chọn $v^{(t+1)}$

$$\alpha_{pq}(v^{(t+1)} | v^{(t-1)}, v^{(t)}) = \begin{cases} \frac{1}{p} & \text{if } \text{dis}(v^{(t-1)}, v^{(t+1)}) = 0 \\ 1 & \text{if } \text{dis}(v^{(t-1)}, v^{(t+1)}) = 1 \\ \frac{1}{q} & \text{if } \text{dis}(v^{(t-1)}, v^{(t+1)}) = 2 \end{cases} \quad \text{dis: độ dài đường đi ngắn nhất}$$

- p điều khiển việc quay lại các đỉnh đã đi qua: p càng nhỏ sẽ càng khuyến khích việc quay lại các đỉnh đã đi qua
- q điều khiển tính hướng nội và hướng ngoại của walk: q càng lớn sẽ càng khuyến khích việc đi tới các đỉnh gần với v^{t-1}

Variants of DeepWalk

- LINE (Tang et al., 2015)
 - Sử dụng các walk với độ dài 1
 - Sử dụng negative sampling

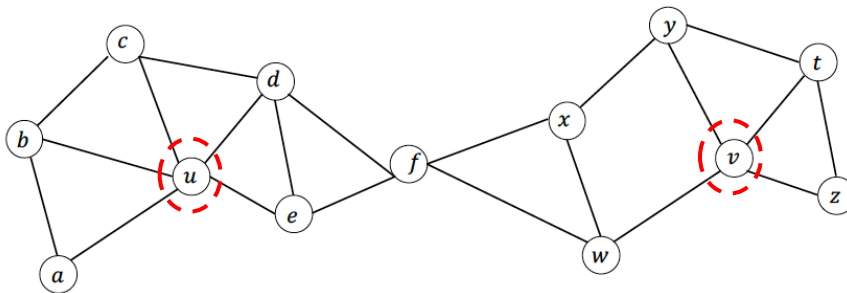
$$- \sum_{(v_{con}, v_{cen}) \in \mathcal{E}} (\log \sigma(f_{con}(v_{con})^\top f_{cen}(v_{cen}))) \\ + \sum_{i=1}^k E_{v_n \sim P_n(v)} [\log \sigma(-f_{con}(v_n)^\top f_{cen}(v_{cen}))],$$


Sử dụng \mathcal{E} (tập các cạnh của đồ thị)

→ là trường hợp đặc biệt của **I** khi chúng ta chỉ tạo các walk độ dài 1

Bảo toàn Structural Role

- Co-occurrence: định nghĩa độ tương đồng của các đỉnh dựa trên khoảng cách
 - Đỉnh càng gần thì càng có khả năng xuất hiện cùng nhau trong 1 walk \rightarrow càng có Co-occurrence cao
- Thực tế, trong 1 số trường hợp độ tương đồng của các đỉnh được thể hiện qua cấu trúc của đồ thị xung quanh nó



u và v có độ tương đồng cao

Ví dụ: phân loại các thành phố

\rightarrow Những thành phố được kết nối với nhiều thành phố khác được coi là có vai trò tương tự nhau, mặc dù chúng có thể cách xa nhau trong đồ thị

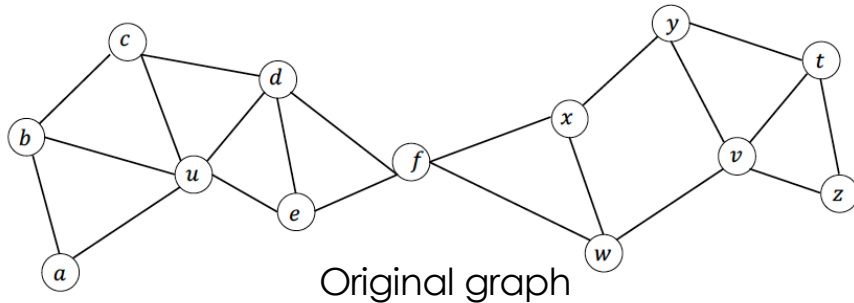
Vd: HN và TPHCM có độ tương đồng cao hơn HN và Bắc Ninh

\rightarrow Co-occurrence không phản ánh được mục tiêu phân loại này

Bảo toàn Structural Role

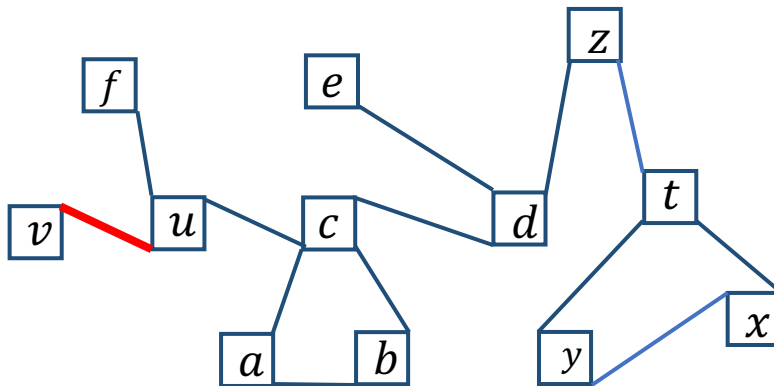
- Struc2vec (Ribeiro et al., 2017):
 - Sử dụng mapping function tương tự như DeepWalk
- Information extractor:
 - Trích xuất structural information dựa trên bậc của các đỉnh
 - Ý tưởng chính:
 - Hai đỉnh có bậc càng giống nhau thì càng có độ tương đồng cao về mặt cấu trúc (high structural similarity)
 - Hai đỉnh mà các láng giềng của chúng có độ tương đồng cấu trúc cao thì chúng càng có độ tương đồng cấu trúc cao
- Flow of Struc2vec
 - Định nghĩa **cách tính Structural Role Similarity**
 - Xây dựng một **đồ thị có trọng số mới** với trọng số của các cạnh thể hiện structural role similarity
 - Áp dụng thuật toán **walk-based trên đồ thị mới**

Struc2vec



1. đồ thị ban đầu

2. Tính structural similarity giữa các đỉnh



3. Xây dựng Structural graph

Là 1 đồ thị có trọng số, với trọng số của các cạnh thể hiện structural similarity giữa các đỉnh

4. Áp dụng DeepWalk trên structural graph

Struc2vec (Ribeiro et al., 2017)

- Định nghĩa Structural Role Similarity

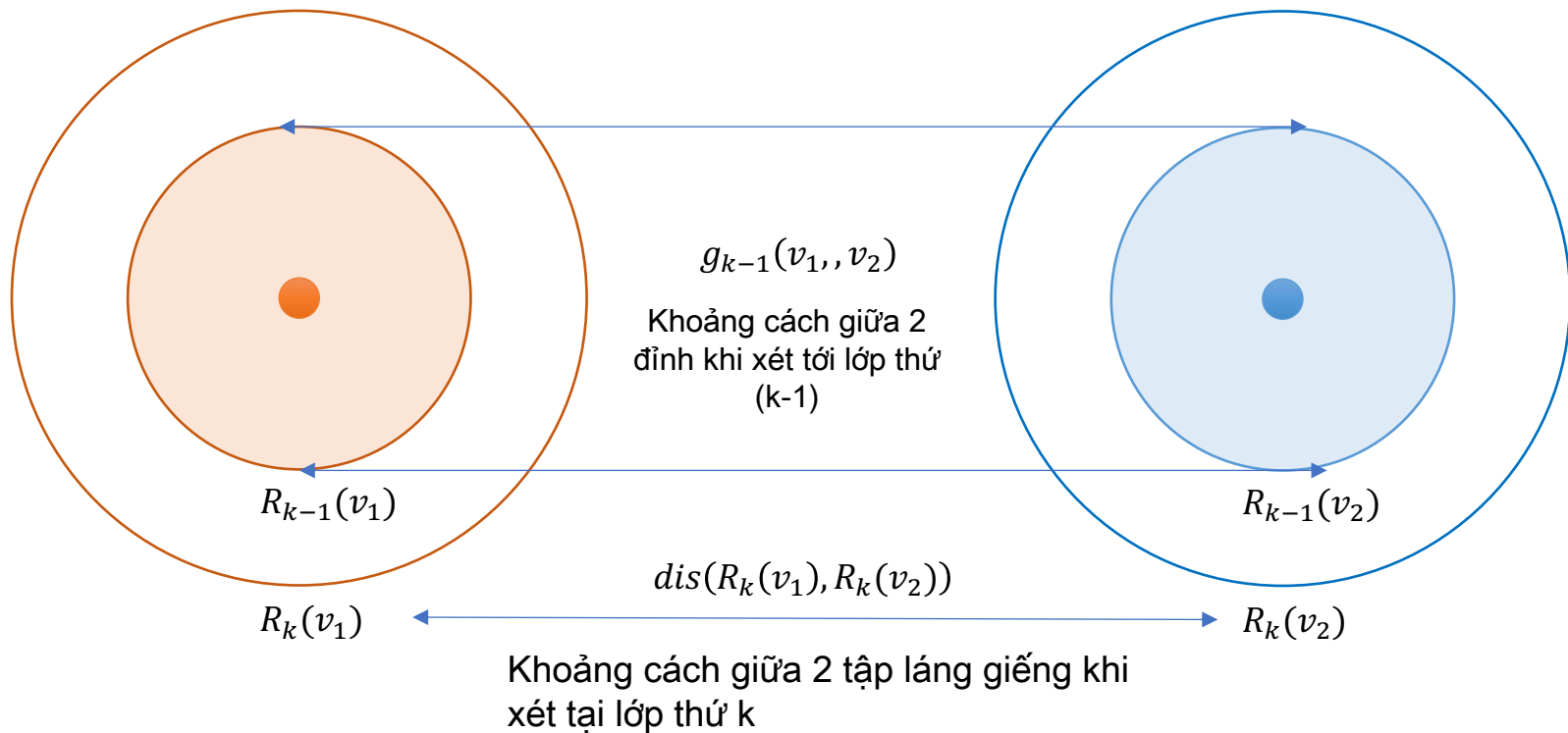
- $R_k(v)$: là tập các đỉnh cách đỉnh v đúng k đỉnh;
 $s(R_k(v))$: $R_k(v)$ sau khi được sắp xếp theo bậc của các đỉnh
- $g_k(v_1; v_2)$: structural difference giữa hai đỉnh v_1 và v_2 khi xem xét các láng giềng cách chúng không quá k đỉnh

$$g_k(v_1, v_2) = g_{k-1}(v_1, v_2) + \text{dis}(s(R_k(v_1)), s(R_k(v_2))),$$



Dynamic Time Warping (DTW) distance:
Đo khoảng cách giữa 2 chuỗi

Struc2vec (Ribeiro et al., 2017)



Struc2vec (Ribeiro et al., 2017)

- Xây dựng structural graph

- **structural graph**: là một đồ thị nhiều lớp, biểu diễn độ tương đồng về cấu trúc (structural similarity) của các đỉnh trong đồ thị ban đầu
- Số lượng lớp: k^* (đường kính của đồ thị ban đầu)
- Trọng số của cạnh (u, v) tại lớp thứ k :

$$w_k(u, v) = \exp(-g_k(u, v))$$

↑
Khoảng cách càng nhỏ, trọng số càng lớn

- Kết nối giữa các lớp:

- Mỗi đỉnh biểu diễn nút v tại lớp thứ k được biểu diễn với đỉnh biểu diễn cùng nút đó tại lớp $k - 1$ và $k + 1$.

Struc2vec (Ribeiro et al., 2017)

- Kết nối giữa các lớp:

$$w(v^{(k)}, v^{(k+1)}) = \log(\Gamma_k(v) + e), k = 0, \dots, k^* - 1,$$

$$w(v^{(k)}, v^{(k-1)}) = 1, k = 1, \dots, k^*,$$

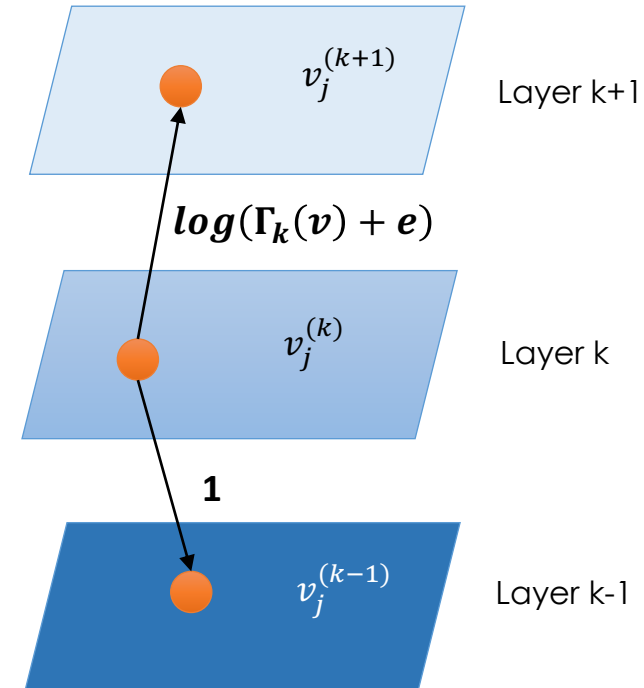
$$\Gamma_k(v) = \sum_{v_j \in \mathcal{V}} \mathbb{1}(w_k(v, v_j) > \bar{w}_k)$$

↑
Thể hiện độ tương đồng về cấu trúc của đỉnh v và các đỉnh khác tại lớp thứ k

↘
Trọng số trung bình của tất cả các cạnh ở lớp thứ k

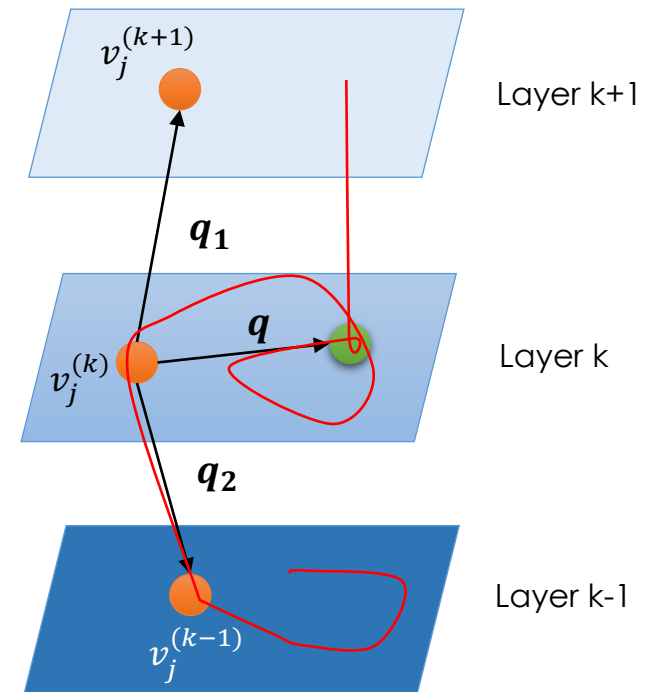
Ý nghĩa của cạnh nối giữa các lớp:

- Trọng số của cạnh càng lớn thì cạnh càng có xác suất được viếng thăm khi thực hiện thuật toán walk
- Một đỉnh tại lớp k sẽ có kết nối mạnh lên lớp $k+1$ nếu nó có độ tương đồng cao với các đỉnh khác ở trong lớp k
- Nói cách khác, nếu một đỉnh không có độ tương đồng với các đỉnh khác tại lớp k thì tại lớp $k+1$ mức độ tương đồng của nó với các đỉnh khác càng yếu hơn vì vậy không khuyến khích walk tiếp lên layer tiếp theo



Struc2vec (Ribeiro et al., 2017)

- Xây dựng structural graph
 - Thực hiện thuật toán Biased Random Walks
 - Giả sử thuật toán walk đang đứng tại đỉnh u của lớp thứ k
 - Tại bước tiếp theo:
 - Ở đi tiếp tới 1 đỉnh thuộc lớp k với xác suất q
 - Chuyển sang lớp trên/hoặc dưới với xác suất $1 - q$
 - q là một hyper parameter



Struc2vec (Ribeiro et al., 2017)

- Xây dựng structural graph
 - Biased Random Walks
 - Xác suất di chuyển từ đỉnh u tới đỉnh v trong cùng 1 lớp

$$p_k(v|u) = \frac{\exp(-g_k(v, u))}{Z_k(u)}, \quad Z_k(u) = \sum_{(v,u) \in \mathcal{E}_k} \exp(-g_k(v, u)).$$

- Xác suất di chuyển tới lớp $k + 1$ và lớp $k - 1$

$$p_k(u^{(k)}, u^{(k+1)}) = \frac{w(u^{(k)}, u^{(k+1)})}{w(u^{(k)}, u^{(k+1)}) + w(u^{(k)}, u^{(k-1)})}$$
$$p_k(u^{(k)}, u^{(k-1)}) = 1 - p_k(u^{(k)}, u^{(k+1)})$$

- Chỉ thu thập các đoạn walk trong cùng 1 lớp

Bảo toàn Node Status

- Node status = global status ranking+ co-occurrence (Ma et al., 2017)
 - Global status ranking: thứ tự xếp hạng của các đỉnh, được đánh giá dựa trên bậc của đỉnh
- Extractor:
 - Tính bậc của các đỉnh
 - Sắp xếp các đỉnh theo độ lớn giảm dần của bậc
 - Thông tin cần bảo toàn: thứ tự xếp hạng của các đỉnh: $(v_{(1)}, \dots, v_{(N)})$
- Objective:
 - Thứ tự tương đối giữa các đỉnh
 - Đỉnh $v_{(i)}$ đứng trước $v_{(j)}$ trong đồ thị ban đầu, thì biểu diễn vector tương ứng của chúng cũng giữ nguyên được quan hệ thứ tự đấy

Bảo toàn Node Status

- Objective (tiếp):

- Bảo toàn thứ tự xếp hạng $(v_{(1)}, \dots, v_{(N)})$
- $p(v_{(i)}; v_{(j)})$ là xác suất để đỉnh $v_{(i)}$ có thứ tự xếp hạng cao hơn $v_{(j)}$, biết rằng biểu diễn vector của $v_{(i)}, v_{(j)}$ là $u_{(i)}, u_{(j)}$

Là trọng số cần học

$$p(v_{(i)}, v_{(j)}) = \sigma(\mathbf{w}^T(\mathbf{u}_{(i)} - \mathbf{u}_{(j)})),$$

$$p_{global} = \prod_{1 \leq i < j \leq N} p(v_{(i)}, v_{(j)}),$$

$$\mathcal{L}_{global} = -\log p_{global}$$

Bảo toàn Node-oriented Structure

- (Wang et al., 2017c) : bảo toàn
 - Thông tin kết nối trực tiếp giữa các đỉnh (Pairwise connectivity information)
 - Độ tương đồng về tập láng giềng của các đỉnh (Similarity between the neighborhoods of nodes)
- Extractor
 - Thông tin kết nối trực tiếp giữa các đỉnh: biểu diễn bằng ma trận kề A .
 - Độ tương đồng về tập láng giềng: định nghĩa cách tính Neighborhood similarity

Bảo toàn Node-oriented Structure

- (Wang et al., 2017c) (tiếp)

Độ tương đồng về tập láng giềng của $v_{(i)}$ và $v_{(j)}$

$$s_{i,j} = \frac{\mathbf{A}_i \mathbf{A}_j^T}{\|\mathbf{A}_i\| \|\mathbf{A}_j\|},$$

→ Định nghĩa ma trận biểu diễn thông tin Node-oriented Structure như sau:

Frobenius Norm

$$\mathbf{P} = \mathbf{A} + \eta \cdot \mathbf{S}, \quad \rightarrow \quad \mathcal{L}(\mathbf{W}) = \underbrace{\|\mathbf{P} - \mathbf{W}\mathbf{W}^T\|_F^2}_{\text{Frobenius Norm}}$$

Ý nghĩa của công hàm loss này là gì?

Bảo toàn Node-oriented Structure

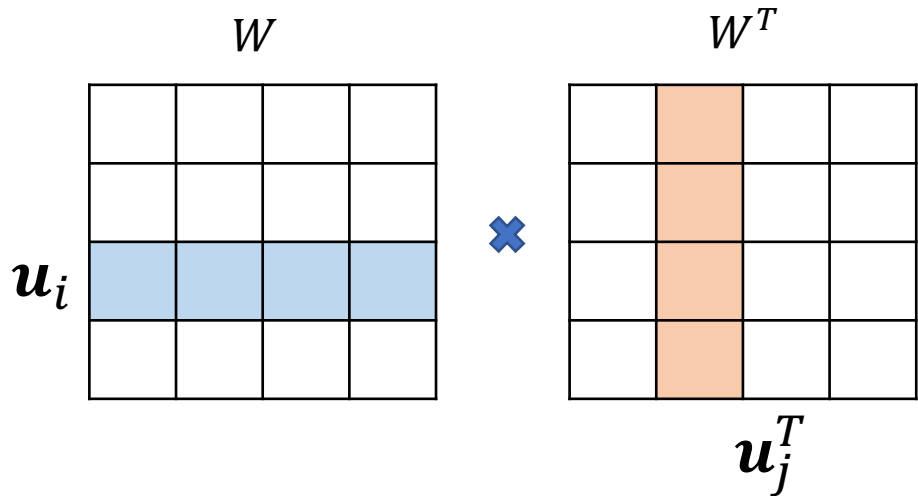
- Ý nghĩa của hàm loss \mathcal{L}
- $f(v_i) = \mathbf{u}_i = \mathbf{e}_i^T \mathbf{W}_{cen}$
- $f(v_j) = \mathbf{u}_j = \mathbf{e}_j^T \mathbf{W}_{con}$

$$s_{i,j} = \frac{\mathbf{A}_i \mathbf{A}_j^T}{\|\mathbf{A}_i\| \|\mathbf{A}_j\|},$$

$$\mathbf{P} = \mathbf{A} + \eta \cdot \mathbf{S},$$

$$\mathcal{L}(\mathbf{W}) = \|\mathbf{P} - \mathbf{W} \mathbf{W}^T\|_F^2$$

$$\mathcal{L}(\mathbf{W}) = \sqrt{\sum_i \sum_j (P_{ij} - \mathbf{u}_i \mathbf{u}_j^T)^2}$$



Độ tương đồng của 2 vector v_i, v_j

Độ tương đồng của 2 đỉnh v_i, v_j trong miền đồ thị

Graph embedding đối với đồ thị không đồng nhất

- Đồ thị không đồng nhất (Heterogeneous graph)
 - Có nhiều loại đỉnh khác nhau
 - Mỗi loại đỉnh có một loại tập miêu tả đặc trưng khác nhau (images, text, ...)

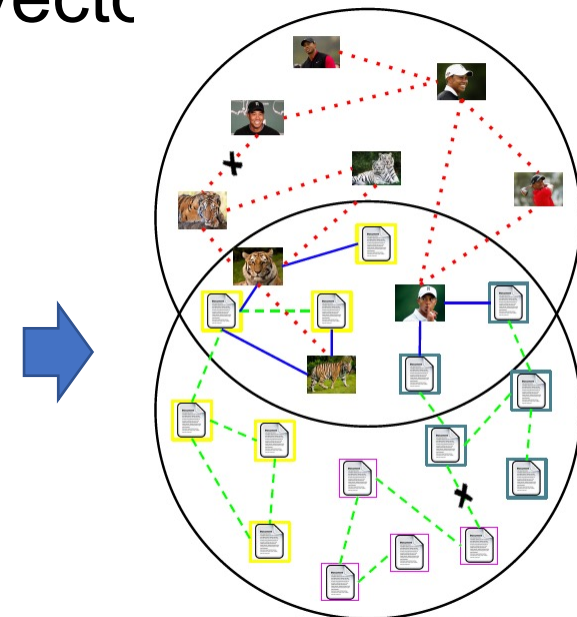
→ sử dụng các model khác nhau để chuyển các loại đặc trưng về cùng 1 không gian vector

Graph embedding đối với đồ thị không đồng nhất

- Question: làm sao để chuyển các loại dữ liệu khác nhau về cùng 1 không gian vectơ



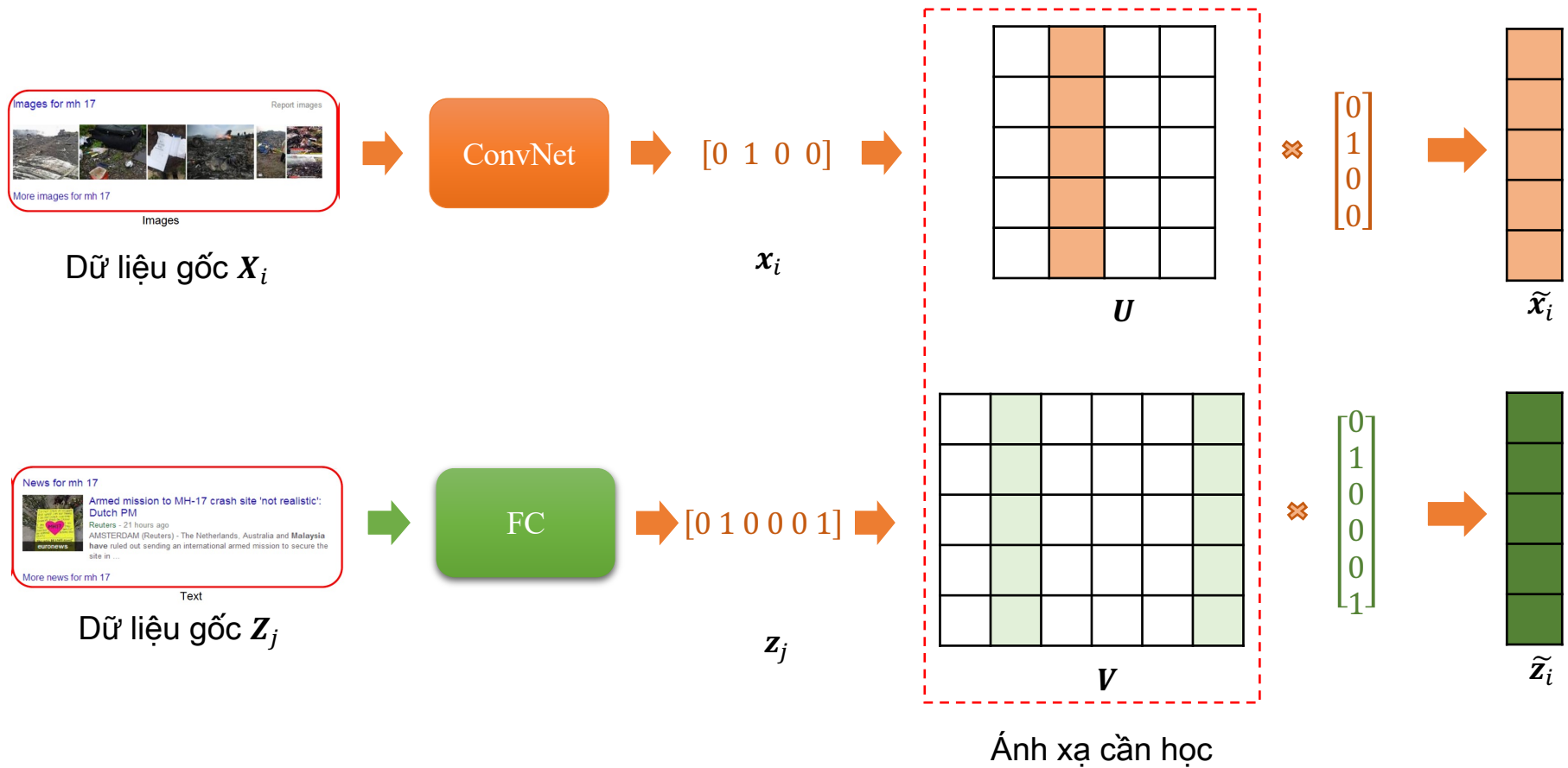
Dữ liệu gốc



Biểu diễn dưới dạng đồ thị

HNE: Heterogeneous Network Embedding via Deep Architectures, KDD'15

Graph embedding đối với đồ thị không đồng nhất



Graph embedding đối với đồ thị không đồng nhất

- Objective: bảo toàn tính liên kề (adjacency)
- Question: nên thiết kế hàm loss như thế nào?
- **Reconstructor:**
 - Giả sử \tilde{x}_i, \tilde{z}_j là các biểu diễn của x_i, z_j trong miền vector
 - \rightarrow định nghĩa xác suất để tồn tại link giữa x_i, z_j như thế nào? \rightarrow có thể coi là 1 bài toán binary classification

Graph embedding đối với đồ thị không đồng nhất

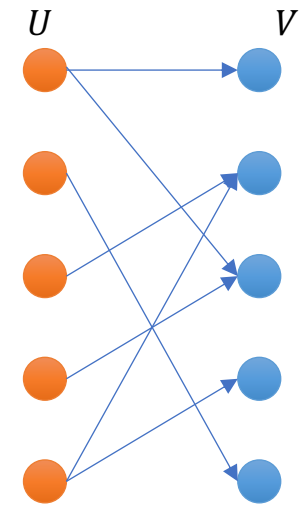
- **Reconstructor:**

- Giả sử \tilde{x}_i, \tilde{z}_j là các biểu diễn của x_i, z_j trong miền vector
- \rightarrow định nghĩa xác suất để tồn tại link giữa x_i, z_j như thế nào? \rightarrow có thể coi là 1 bài toán binary classification
- $\rightarrow p(\tilde{A}_{ij} = 1) = \sigma(\tilde{x}_i^T \cdot \tilde{z}_j)$ \leftarrow Maximize nếu $A_{ij} = 1$
- $\rightarrow p(\tilde{A}_{ij} = 0) = 1 - \sigma(\tilde{x}_i^T \cdot \tilde{z}_j)$ \leftarrow Maximize nếu $A_{ij} = 0$

$$\begin{aligned}\mathcal{L} &= -\log\left(\prod_{A_{ij}=1} p(\tilde{A}_{ij} = 1) \times \prod_{A_{ij}=0} p(\tilde{A}_{ij} = 0)\right) \\ &= -\sum_{\forall i,j} A_{ij} \log p(\tilde{A}_{ij} = 1) + (1 - A_{ij}) \log p(\tilde{A}_{ij} = 0)\end{aligned}$$

Bipartite graph Embedding

- Input: đồ thị hai phía (bipartite network):
 $G = (U, V, E)$, ma trận trọng số W
- Output: ánh xạ $f: U \cup V \rightarrow \mathbb{R}^d$, ánh xạ mỗi đỉnh sang 1 vector
- Thông tin cần bảo toàn
 - Lân cận bậc 1 (1st-order proximity): các đỉnh liền kề
 - Đại diện cho quan hệ giữa 2 tập đỉnh
 - Lân cận bậc 2: các đỉnh cách 2 hop
 - Đại diện cho quan hệ giữa các đỉnh trong cùng 1 loại



Đồ thị hai phía

Không có cạnh giữa các đỉnh thuộc cùng 1 loại

Bipartite graph Embedding

- Mô hình hoá quan hệ lân cận bậc 1

- **joint probability** giữa hai đỉnh u_i và v_j :

$$P(i, j) = \frac{w_{ij}}{\sum_{e_{ij} \in E} w_{ij}}$$

w_{ij} là trọng số của cạnh $e_{ij} \rightarrow$ trọng số càng cao, joint probability càng lớn
 $\rightarrow P(i, j)$ thể hiện quan hệ của 2 đỉnh kề nhau

- Mỗi quan hệ của 2 đỉnh dựa trên giá trị của các vector tương ứng của chúng

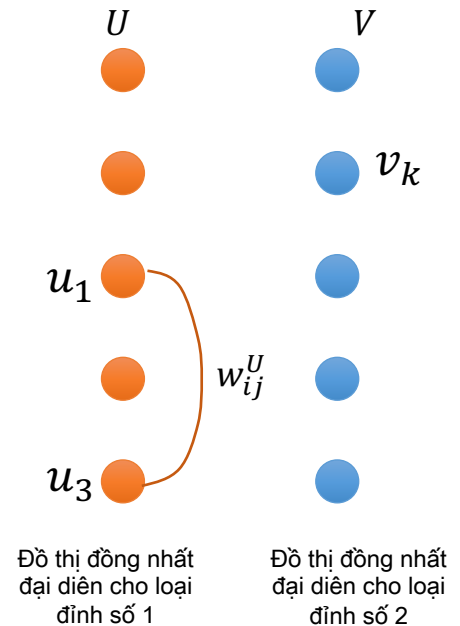
$$\hat{P}(i, j) = \frac{1}{1 + e^{-(\vec{u}_i^T \cdot \vec{v}_j)}}$$

\rightarrow Hàm loss biểu diễn khả năng bảo toàn thông tin về quan hệ lân cận bậc 1 được định nghĩa bởi:

$$\begin{aligned} O_1 = KL(P || \hat{P}) &= \sum_{e_{ij} \in E} P(i, j) \log \frac{P(i, j)}{\hat{P}(i, j)} = \sum_{e_{ij} \in E} \frac{w_{ij}}{\sum_{e_{ij} \in E} w_{ij}} \left(\log \frac{w_{ij}}{\sum_{e_{ij} \in E} w_{ij}} - \log \hat{P}(i, j) \right) \\ &\propto - \sum_{e_{ij} \in E} w_{ij} \log \hat{P}(i, j) \end{aligned}$$

Bipartite graph Embedding

- Mô hình hoá quan hệ lân cận bậc 2
 - Xây dựng 2 đồ thị đồng nhất (homogeneous graph), mỗi đồ thị chứa các đỉnh của cùng 1 loại
 - Các cạnh biểu diễn quan hệ lân cận bậc 2
 - Tạo walk trên 2 đồ thị đồng nhất này
 - Question: định nghĩa quan hệ lân cận bậc 2 như thế nào?



$$w_{ij}^U = \sum_{k \in V} w_{ik} w_{jk}; \quad w_{ij}^V = \sum_{k \in U} w_{ki} w_{kj}.$$

Bipartite graph Embedding

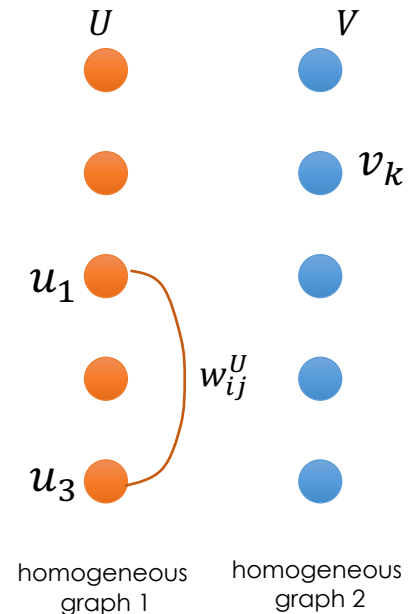
- Mô hình hoá quan hệ lân cận bậc 2
 - Các cạnh của 2 đồ thị đồng nhất được gán trọng số W_{ij}^U, W_{ij}^V
 - Áp dụng thuật toán biasedWalk trên 2 đồ thị đồng nhất
 - Số lượng walk xuất phát từ đỉnh $v_i = \max(H(v_i) \times \max T, \min T)$
 - Định nghĩa hàm Loss nhằm tối đa hoá việc việc toàn quan hệ lân cận bậc 2

$$\text{maximize } O_2 = \prod_{u_i \in S \wedge S \in D^U} \prod_{u_c \in C_S(u_i)} P(u_c | u_i).$$

$$\text{maximize } O_3 = \prod_{v_j \in S \wedge S \in D^V} \prod_{v_c \in C_S(v_j)} P(v_c | v_j).$$

$$P(u_c | u_i) = \frac{\exp(\vec{u}_i^T \vec{\theta}_c)}{\sum_{k=1}^{|U|} \exp(\vec{u}_i^T \vec{\theta}_k)}, \quad P(v_c | v_j) = \frac{\exp(\vec{v}_j^T \vec{\theta}_c)}{\sum_{k=1}^{|V|} \exp(\vec{v}_j^T \vec{\theta}_k)}.$$

$$\text{maximize } L = \alpha \log O_2 + \beta \log O_3 - \gamma O_1.$$



Tại sao đây lại là dấu -?

Mạng nơon đồ thị

Graph neural network (GNN)

Graph neural network

- Định nghĩa
 - Là các mạng nơon mà dữ liệu đầu vào là dạng đồ thị
- Các dạng bài toán
 - Node-focused tasks: là các bài toán mà mỗi điểm dữ liệu tương ứng với 1 đỉnh của đồ thị
 - Graph-based tasks: là các dạng dữ liệu mà mỗi điểm dữ liệu tương ứng với 1 đồ thị

Mô hình GNN tổng quát

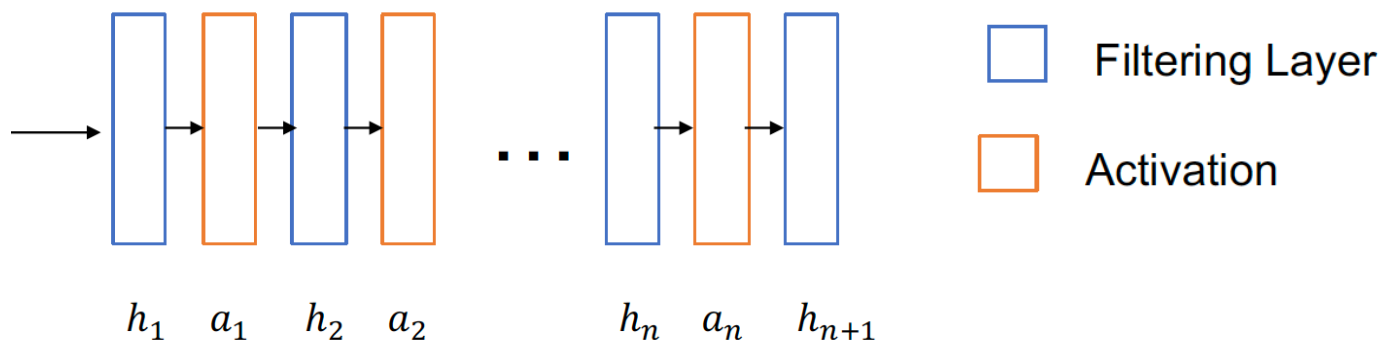
- Đối với Node-focused tasks:
 - Mạng GNN thường bao gồm nhiều lớp graph filtering + activation
- Đối với Graph-based tasks:
 - Mạng GNN thường bao gồm nhiều lớp graph filtering + activation layer và theo sau bởi lớp graph pooling
- Lớp Graph filtering và activation layers
 - Trích xuất các đặc trưng có giá trị của các đỉnh
 - Đầu ra của các lớp này là các đồ thị mới có cùng cấu trúc (cùng đỉnh, cùng cạnh)
- Lớp Graph pooling
 - Tổng hợp thông tin của tất cả các đỉnh và cạnh để trích xuất thông tin tổng quan của toàn bộ đồ thị
 - Output là các thông tin ở mức trừu tượng cao hơn so với output
→ output thường có số đỉnh, số cạnh khác input

Mô hình GNN tổng quát

- Node-focused GNN
 - Lớp graph filtering thứ i

$$\mathbf{F}^{(i)} = h_i \left(\mathbf{A}, \alpha_{i-1}(\mathbf{F}^{(i-1)}) \right)$$

Ma trận kề của đồ thị ban đầu
→ Không thay đổi khi qua các layer



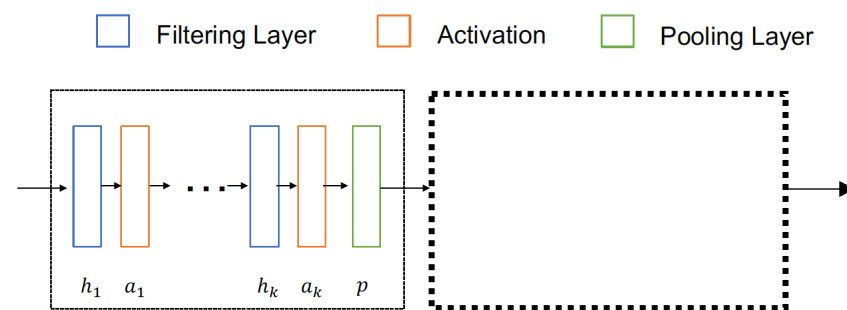
Là một chuỗi các lớp graph **filtering** và **activation** layers

GNN for node-focused tasks

Mô hình GNN tổng quát

- Graph-based tasks: Cấu tạo bởi nhiều khối liên tiếp, mỗi khối có:

- Input: đồ thị $G_{ib}\{V_{ib}, E_{ib}\}$
 - Ma trận kề $A^{(ib)}$
 - Đặc trưng của các đỉnh: $F^{(ib)}$
- Output: đồ thị $G_{ob}\{V_{ob}, E_{ob}\}$
 - Ma trận kề $A^{(ob)}$
 - Đặc trưng của các đỉnh: $F^{(ob)}$



Mỗi block bao gồm **nhiều lớp** graph filtering + activation layers và **một lớp** pooling

GNN for graph-focused tasks

$$F^{(i)} = h_i \left(\boxed{A^{(ib)}}, \alpha_{i-1}(F^{(i-1)}) \right) \text{ for } i = 1, \dots, k$$

$$A^{(ob)}, F^{(i)} = p \left(\boxed{A^{(ib)}}, F^{(k)} \right)$$

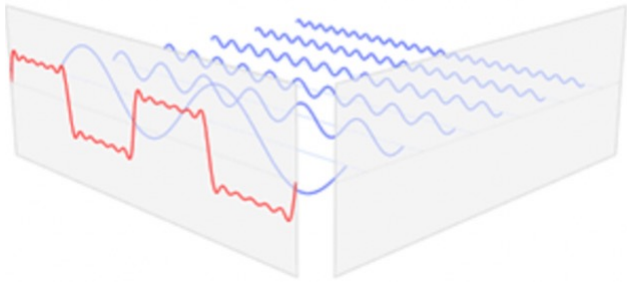
Ma trận kề nhận từ khối liên trước
 → Không thay đổi trong 1 block
 → Thay đổi khi đi qua các block

Graph Filters

- Các nhóm giải pháp
 - Spatial-based graph filters
 - Sử dụng các thông tin tường minh về cấu trúc đồ thị (vd: kết nối giữa các đỉnh, số bậc của các đỉnh,) để trích xuất các thông tin có ý nghĩa của đồ thị
 - Spectral-based Graph Filters
 - Sử dụng lý thuyết về đồ thị quang phổ (spectral graph theory)
 - Thực hiện việc filter trên miền tần số
- Thông thường chúng ta quen hơn với nhóm giải pháp đầu
- Tuy nhiên, rất nhiều giải pháp thuộc nhóm đầu được xây dựng từ lý thuyết của nhóm giải pháp thứ 2

Spectral-based Graph Filters

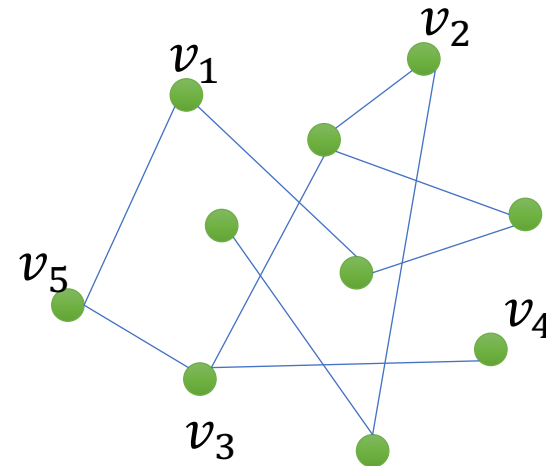
- Dựa trên lý thuyết về đồ thị quang phổ



Nhắc lại về biến đổi Fourier đối với hàm liên tục

$$f(t) = \int_{-\infty}^{\infty} \hat{f}(\xi) e^{2\pi i t \xi} d\xi$$

$e^{2\pi i t \xi} = \cos(2\pi t \xi) + i \sin(2\pi t \xi)$



Tập thông tin của các đỉnh thuộc 1 đồ thị có thể biểu diễn dưới dạng chuỗi $\{f(v_1), \dots, f(v_n)\}$,

Trong đó, $f(v_i)$ là thông tin của đỉnh v_i

$f = \{f(v_1), \dots, f(v_n)\}$ được gọi là tín hiệu đồ thị (**graph signal**)

→ Tần số = sự thay đổi của các giá trị của tín hiệu f khi ta di chuyển trên các cạnh

→ giá trị của f ít thay đổi qua các cạnh, nghĩa là tần số của f thấp

→ Biến đổi Fourier trên miền đồ thị là việc phân tách f thành tổ hợp tuyến tính các tín hiệu thành phần, mỗi tín hiệu thành phần là có một tần số nhất định nào đó

Nhắc lại về giải tích và đại số tuyến tính

- Toán tử Laplacian, Ma trận Laplacian
- Eigenfunctions
- Biến đổi Fourier

Toán tử Laplacian

- Toán tử Laplacian

- Tích phân bậc 1 (toán tử Nabla): $\nabla f = \frac{\partial f}{\partial x}$
- Tích phân bậc 2 (toán tử Laplacian): $\Delta f = \frac{\partial^2 f}{\partial x^2}$

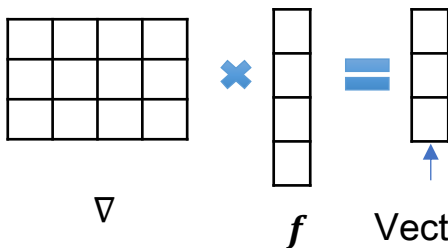
- Ma trận Laplacian

f là graph signal

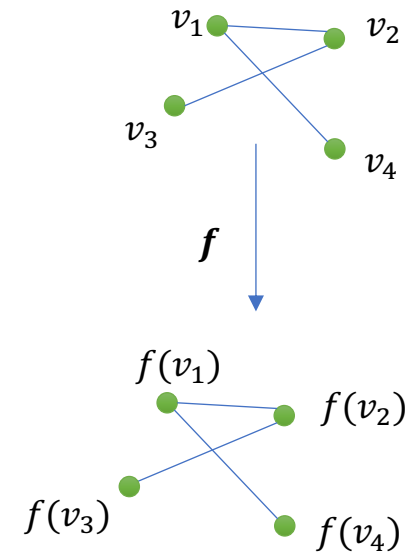
Khái niệm tương tự của tích phân bậc 1 trong miền đồ thị

$$g = \nabla f$$

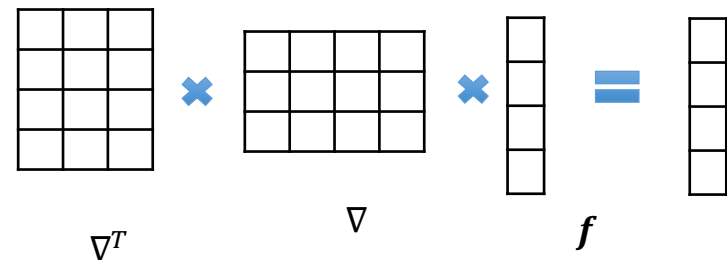
$$g(e_{ij}) = f(v_i) - f(v_j)$$



Mỗi phần tử tương ứng với 1 cạnh



$$Lf(v_i) = \nabla^T \nabla f = \sum_{v_j \in N(v_i)} f(v_i) - f(v_j)$$

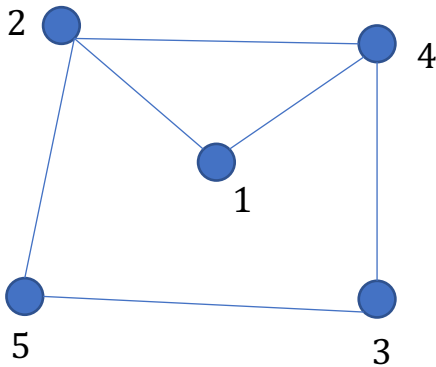


Ma trận Laplacian

- $$Lf(v_i) = \sum_{v_j \in N(v_i)} f(v_i) - f(v_j)$$

$$= |N(v_i)| \cdot f(v_i) - \sum_{\forall v_j} A_{ij} \cdot f(v_j)$$

$$= (D - A)f(v_i)$$



$D(v_2)$ →

2				
	3			
		3		
			2	
				2

D

0	1		1	
1	0		1	1
		0	1	1
1	1	1	0	
	1	1		0

$A(v_2)$

A

Ma trận Laplacian

- $Lf(v_i) = \sum_{v_j \in N(v_i)} f(v_i) - f(v_j)$
- $f^T Lf = \frac{1}{2} \sum_{v_i \in V} \sum_{v_j \in N(v_i)} (f(v_i) - f(v_j))^2$
- Độ mượt của đồ thị (Graph smoothness): 1 đồ thị càng mượt nếu giá trị của các đỉnh của nó ít thay đổi
 - Đồ thị càng mượt \rightarrow sự sai khác giữa các đỉnh càng nhỏ $\rightarrow f^T Lf$ càng nhỏ
 - $f^T Lf$ có thể dùng để biểu diễn độ mượt của graph-signal f

Biến đổi Fourier trong miền đồ thị

- $\hat{f}(\xi) = \langle f(t), \exp(-2\pi i t \xi) \rangle = \int_{-\infty}^{\infty} f(t) \exp(-2\pi i t \xi) dt$
 - $\hat{f}(\xi)$ là hệ số tương ứng với thành phần $\exp(-2\pi i t \xi)$
 - $f(t) = \int_{-\infty}^{\infty} \hat{f}(\xi) e^{2\pi i t \xi} d\xi$
- Tại sao lại phân tách tín hiệu dựa trên hàm số $\exp(-2\pi i t \xi)$?
 - $\exp(-2\pi i t \xi)$ là các eigenfunctions của toán tử Laplacian
 - $\nabla \exp(-2\pi i t \xi) = -(2\pi i \xi)^2 \exp(-2\pi i t \xi)$
- Nhắc lại về eigenfunction
 - Giả sử D là 1 toán tử: $Df = g$
 - f^* là eigenfunction của D nếu $Df^* = \lambda f^*$

Biến đổi Fourier trong miền đồ thị

- Biểu diễn 1 graph signal dưới dạng tổ hợp tuyến tính của các eigenvectors của ma trận Laplacian L
 - Nhắc lại biến đổi Fourier trong miền hàm liên tục
 - $\hat{f}(\xi) = \langle f(t), \exp(-2\pi i t \xi) \rangle$;
 - $\exp(-2\pi i t \xi)$: eigenfunction của toán tử Laplacian
 - \rightarrow ánh xạ sang miền đồ thị:
 - $\hat{f}[l] = \langle \mathbf{f}, \mathbf{u}_l \rangle = \sum_{i=1}^N \mathbf{f}[i] \mathbf{u}_l[i]$
 - \mathbf{u}_l : là eigenvector thứ l của ma trận L
 - λ_l : là độ mượt của tín hiệu \mathbf{u}_l
 - $\mathbf{u}_l^T L \mathbf{u}_l = \mathbf{u}_l^T (L \mathbf{u}_l) = \mathbf{u}_l^T (\lambda_l \mathbf{u}_l) = \lambda_l \mathbf{u}_l^T \mathbf{u}_l = \lambda_l$
 - λ_l có ý nghĩa như tần số của tín hiệu \mathbf{u}_l
 - Tại sao?

Biến đổi Fourier trong miền đồ thị

- U : ma trận có các cột là các eigenvectors của ma trận Laplacian L

$$\hat{f}[l] = \langle f, u_l \rangle = \sum_{i=1}^N f[i] u_l[i]$$

$$\rightarrow \hat{f} = U^T f$$

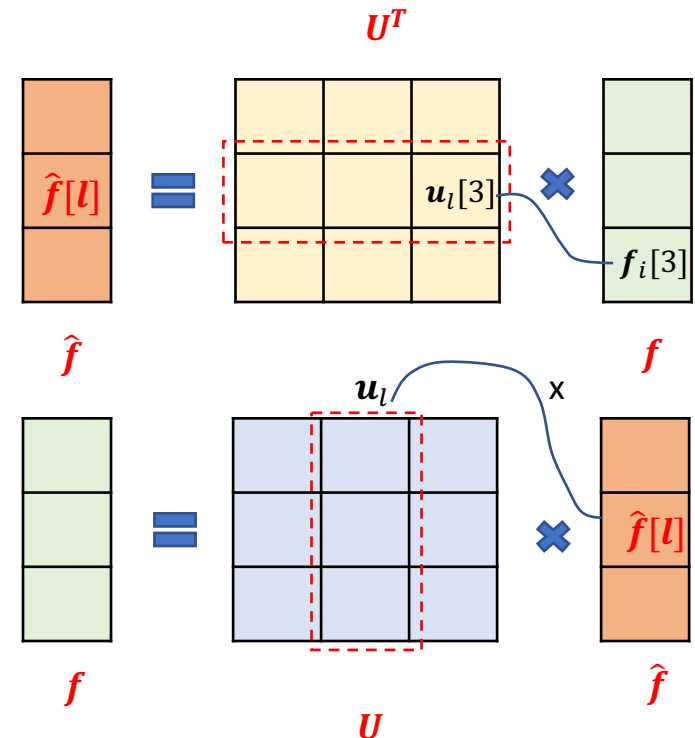
$$\rightarrow U \hat{f} = U(U^T f) = (UU^T)f = f$$

$$\rightarrow f = U \hat{f}$$

$$\rightarrow f = \hat{f}[1]u_1 + \dots + \hat{f}[m]u_m$$

Hệ số tương ứng

1 tín hiệu thành phần



Biến đổi Fourier trong miền đồ thị

- Trong miền đồ thị

$$f = \hat{f}[1]u_1 + \dots + \hat{f}[m]u_m$$

Hệ số tương ứng

Mỗi thành phần = **eigenvector** của ma trận **Laplacian**

- Trong miền hàm số liên tục

$$f(t) = \int_{-\infty}^{\infty} \hat{f}(\xi) e^{2\pi i t \xi} d\xi$$

Hệ số tương ứng

Mỗi thành phần = an **eigenfunction** của toán tử **Laplacian operator**

Spectral-based Graph Filters

- Ý tưởng

- Điều chế các tần số của graph signal đầu vào, nhằm giữ lại các thành phần với tần số quan trọng, loại bỏ những thành phần với tần số không quan trọng

- Flow

- Input: graph signal f ($f[i]$ biểu diễn thông tin của đỉnh v_i)
 - Thực hiện biến đổi Fourier $\hat{f} = U^T f$
 - Thực hiện Filter trên \hat{f}
 - $\hat{f}'[i] = \hat{f}[i] \cdot \gamma(\lambda_i)$
 - Thực hiện biến đổi Fourier ngược
 - $f' = U \hat{f}'$
- Ma trận bao gồm tất cả các eigenvector của ma trận Laplacian L
- Hàm số xác định nên tăng/giảm thành phần của tần số nào.
- Đây là cái mô hình của chúng ta cần học

Spectral-based Graph Filters

- $\hat{f}'[i] = \hat{f}[i] \cdot \gamma(\lambda_i) \rightarrow \hat{f}' = \gamma(\Lambda)\hat{f} = \gamma(\Lambda)U^T f$
- $f' = U\hat{f}' = U\gamma(\Lambda)U^T f$

$$\Lambda = \begin{pmatrix} \lambda_1 & & 0 \\ & \ddots & \\ 0 & & \lambda_N \end{pmatrix}; \quad \gamma(\Lambda) = \begin{pmatrix} \gamma(\lambda_1) & & 0 \\ & \ddots & \\ 0 & & \gamma(\lambda_N) \end{pmatrix}. \quad \text{Ví dụ} \quad \gamma(\lambda_l) = \frac{1}{1 + c\lambda_l},$$

→ Đây là một hàm γ đơn giản nhất

Spectral-based Graph Filters

- $f' = U\hat{f}' = U\gamma(\Lambda)U^T f$
- Một số hàm $\gamma(\Lambda)$ khác

$$\gamma(\Lambda) = \begin{pmatrix} \theta_1 & & 0 \\ & \ddots & \\ 0 & & \theta_N \end{pmatrix}. \quad \theta_i \text{ là các tham số cần học}$$

- Poly-Filter

$$\gamma(\lambda_l) = \sum_{k=0}^K \theta_k \lambda_l^k. \quad \Rightarrow \quad \gamma(\Lambda) = \sum_{k=0}^K \theta_k \Lambda^k. \quad \Rightarrow \quad \mathbf{f}' = \sum_{k=0}^K \theta_k \underbrace{\mathbf{U} \cdot \Lambda^k \cdot \mathbf{U}^T}_{\mathbf{L}^k} \mathbf{f} = \sum_{k=0}^K \theta_k \mathbf{L}^k \mathbf{f}.$$

Poly-Filter-based Graph Filters

$$f' = \sum_{k=0}^K \theta_k L^k f$$

$$\rightarrow f'[i] = \sum_j \sum_{k=0}^K \theta_k L_{i,j}^k f[j]$$

Định lý: $L_{ij}^k = 0$ nếu $dis(v_i, v_j) > k$

$$\rightarrow f'[i] = \sum_{v_j \in N^K(v_i) \& v_i} \left[\sum_{k=0}^K \theta_k L_{i,j}^k \right] f[j]$$

$$\rightarrow f'[i] = b_{ii} f[i] + \sum_{v_j \in N^K(v_i)} b_{ij} \cdot f[j]$$

Ý nghĩa: thông tin của đỉnh v_i được tính từ thông tin của chính nó và các đỉnh cách nó 1 khoảng cách không vượt quá K

→ Đây là ý tưởng cơ bản được áp dụng trong nhiều mô hình GNN hiện nay

Spectral-based Graph Filters

- Hạn chế của Poly-Filter'
 - Các đa thức $(1; x; x^2; \dots)$ không trực giao \rightarrow quá trình học các hệ số của đa thức này sẽ ảnh hưởng tới các đa thức khác \rightarrow dẫn tới tính bất ổn định của quá trình huấn luyện
- Giải pháp: sử dụng các đa thức Chebyshev
 - Là hệ trực giao

Spectral-based Graph Filters

- Chebyshev polynomial: là tập hợp các đa thức trực giao

$$T_k(y) = 2yT_{k-1}(y) - T_{k-2}(y), \quad T_0(y) = 1 \text{ and } T_1(y) = y,$$

$$\gamma(\Lambda) = \sum_{k=0}^K \theta_k T_k(\tilde{\Lambda}).$$

$$\mathbf{f}' = \mathbf{U} \cdot \sum_{k=0}^K \theta_k T_k(\tilde{\Lambda}) \mathbf{U}^\top \mathbf{f} = \sum_{k=0}^K \theta_k \mathbf{U} T_k(\tilde{\Lambda}) \mathbf{U}^\top \mathbf{f}.$$

$$\mathbf{U} T_k(\tilde{\Lambda}) \mathbf{U}^\top = T_k(\tilde{\mathbf{L}}) \text{ with } \tilde{\mathbf{L}} = \frac{2\mathbf{L}}{\lambda_{\max}} - \mathbf{I} \quad \Rightarrow \quad \mathbf{f}' = \sum_{k=0}^K \theta_k \mathbf{U} T_k(\tilde{\Lambda}) \mathbf{U}^\top \mathbf{f} = \sum_{k=0}^K \theta_k T_k(\tilde{\mathbf{L}}) \mathbf{f}$$

Đa thức Cheby-Filter vẫn giữ được các ưu điểm của Poly-filter trong khi đảm bảo được tính ổn định của quá trình huấn luyện

Graph convolutional network (GCN)

- Đơn giản hoá của Cheby-Filter

- $K = 1 \rightarrow T_0(y) = 1 ; T_1(y) = y$

- $f' = \sum_{k=0}^K \theta_k T_k(\tilde{L})f = (\theta_0 T_0(\tilde{L}) + \theta_1 T_1(\tilde{L}))f = (\theta_0 I + \theta_1 \tilde{L})f$

- $\tilde{L} = \frac{2L}{\lambda_{max}} - I = L - I$ with λ_{max} is set to 2

- $f' = (\theta_0 I + \theta_1 (L - I))f$

- Setting $\theta_0 = -\theta_1 = \theta$

$$\rightarrow f' = \theta \tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}} f$$

trong GCN: thông tin của 1 đỉnh được tính từ các đỉnh kề nó và chính nó

Thành phần (i, j) chỉ khác 0 nếu 2 đỉnh i, j kề nhau

Graph convolutional network (GCN)

Bài toán: node classification

- Input: 1 đồ thị với 1 số đỉnh đã có nhãn
- Mục tiêu: xác định nhãn cho các đỉnh còn lại
- Phương pháp: 2 lớp filters
 - $\hat{A} = \tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}}$
 - Filter layer 1: $(X, A) \rightarrow \hat{A} X W^{(0)}$
 - Filter layer 2: $\hat{A} X W^{(0)} \rightarrow \hat{A} \times \text{ReLU}(\hat{A} X W^{(0)}) \times W^{(1)}$
 - Output layer: $\hat{A} \times \text{ReLU}(\hat{A} X W^{(0)}) \times W^{(1)} \rightarrow \text{Softmax}(\hat{A} \times \text{ReLU}(\hat{A} X W^{(0)}) \times W^{(1)})$
 - Hàm Loss : cross entropy $\mathcal{L} = - \sum_{l \in Y_L} \sum_{f=1}^F Y_{lf} \ln(Z_{lf})$

↑
label

↖
Output vector

GAT: Graph attention networks

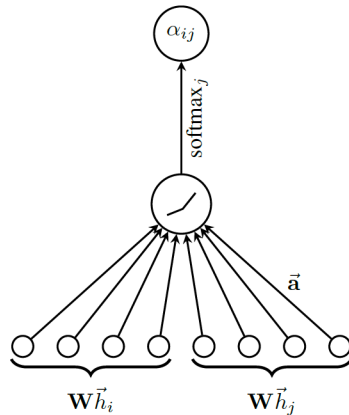
- Sử dụng cơ chế attention để trích xuất thông tin từ các đỉnh, giải quyết bài toán node classification
- Xác định biểu diễn của mỗi đỉnh bằng cách áp dụng cơ chế attention đối với các đỉnh kề của nó

GRAPH ATTENTION NETWORKS, ICLR 2018

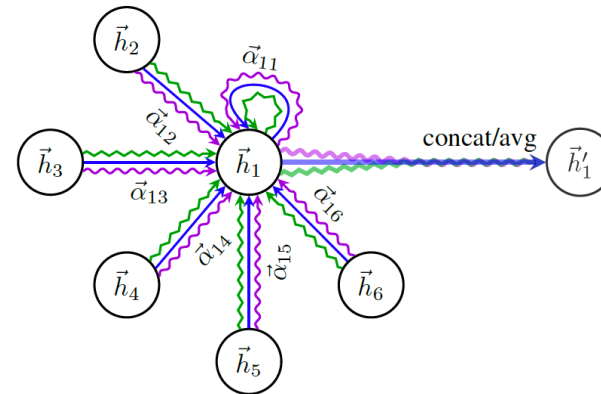
GAT: Graph attention networks

- Ý tưởng chính

- Định nghĩa attention weights với các đỉnh kề (1-hop neighbors)
- Xây dựng nhiều lớp attention, mỗi lớp học một bộ weight khác nhau



Self-attention



Multi-head attention

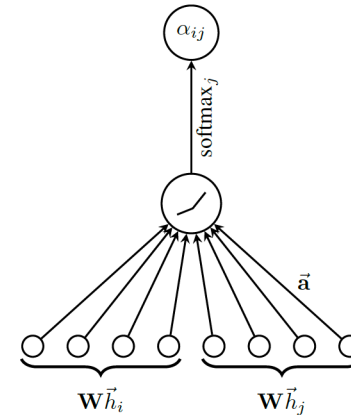
GAT: Graph attention networks

- Self-attention

$$\vec{h}'_i = \sigma \left(\sum_{j \in \mathcal{N}_i} \alpha_{ij} \mathbf{W} \vec{h}_j \right).$$

Vector biểu diễn đỉnh thứ i

Hệ số attention



$$\alpha_{ij} = \text{softmax}_j(e_{ij}) :$$

$$e_{ij} = a(\mathbf{W} \vec{h}_i, \mathbf{W} \vec{h}_j)$$

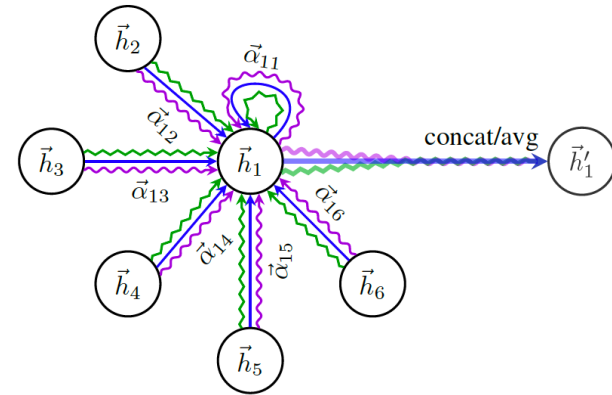
a là single-layer feedforward neural network, theo sau bởi 1 hàm LeakyReLU

$$\alpha_{ij} = \frac{\exp \left(\text{LeakyReLU} \left(\vec{a}^T [\mathbf{W} \vec{h}_i \| \mathbf{W} \vec{h}_j] \right) \right)}{\sum_{k \in \mathcal{N}_i} \exp \left(\text{LeakyReLU} \left(\vec{a}^T [\mathbf{W} \vec{h}_i \| \mathbf{W} \vec{h}_k] \right) \right)}$$

GAT: Graph attention networks

- Multi-head attention

$$\vec{h}'_i = \bigparallel_{k=1}^K \sigma \left(\sum_{j \in \mathcal{N}_i} \alpha_{ij}^k \mathbf{W}^k \vec{h}_j \right)$$



- Trong bài báo, tác giả đề xuất

$$\vec{h}'_i = \sigma \left(\frac{1}{K} \sum_{k=1}^K \sum_{j \in \mathcal{N}_i} \alpha_{ij}^k \mathbf{W}^k \vec{h}_j \right)$$

graphSAGE

- Instead of training a distinct embedding vector for each node, we train a set of aggregator functions
- Each aggregator function aggregates information from a different number of hops, or search depth, away from a given node.

graphSAGE

Algorithm 1: GraphSAGE embedding generation (i.e., forward propagation) algorithm

Input : Graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$; input features $\{\mathbf{x}_v, \forall v \in \mathcal{V}\}$; depth K ; weight matrices $\mathbf{W}^k, \forall k \in \{1, \dots, K\}$; non-linearity σ ; differentiable aggregator functions $\text{AGGREGATE}_k, \forall k \in \{1, \dots, K\}$; neighborhood function $\mathcal{N} : v \rightarrow 2^{\mathcal{V}}$

Output : Vector representations \mathbf{z}_v for all $v \in \mathcal{V}$

```
1  $\mathbf{h}_v^0 \leftarrow \mathbf{x}_v, \forall v \in \mathcal{V}$  ;
2 for  $k = 1 \dots K$  do
3   for  $v \in \mathcal{V}$  do
4      $\mathbf{h}_{\mathcal{N}(v)}^k \leftarrow \text{AGGREGATE}_k(\{\mathbf{h}_u^{k-1}, \forall u \in \mathcal{N}(v)\})$ ;
5      $\mathbf{h}_v^k \leftarrow \sigma \left( \mathbf{W}^k \cdot \text{CONCAT}(\mathbf{h}_v^{k-1}, \mathbf{h}_{\mathcal{N}(v)}^k) \right)$ 
6   end
7    $\mathbf{h}_v^k \leftarrow \mathbf{h}_v^k / \|\mathbf{h}_v^k\|_2, \forall v \in \mathcal{V}$ 
8 end
9  $\mathbf{z}_v \leftarrow \mathbf{h}_v^K, \forall v \in \mathcal{V}$ 
```

graphSAGE

- Loss function

$$J_{\mathcal{G}}(\mathbf{z}_u) = -\log(\sigma(\mathbf{z}_u^\top \mathbf{z}_v)) - Q \cdot \mathbb{E}_{v_n \sim P_n(v)} \log(\sigma(-\mathbf{z}_u^\top \mathbf{z}_{v_n}))$$

- v is a node that co-occurs near u on fixed-length random walk

- **Aggregate function** $\mathbf{h}_v^k \leftarrow \sigma(\mathbf{W} \cdot \text{MEAN}(\{\mathbf{h}_v^{k-1}\} \cup \{\mathbf{h}_u^{k-1}, \forall u \in \mathcal{N}(v)\})).$

- Mean $\text{AGGREGATE}_k^{\text{pool}} = \text{max}(\{\sigma(\mathbf{W}_{\text{pool}} \mathbf{h}_{u_i}^k + \mathbf{b}), \forall u_i \in \mathcal{N}(v)\}),$
- LSTM
- Max pooling

Tổng kết

- Graph embedding
 - Xây dựng ánh xạ từ miền đồ thị sang miền vector
 - Mỗi đỉnh của đồ thị được ánh xạ sang 1 vector
 - Các thành phần cơ bản
 - Mapping function, information extractor, reconstructor, objective
 - Mapping function: table lookup
 - Information extractor: walk-based
- Graph neural network
 - Node-based tasks: graph filter + activation
 - Graph-based tasks : graph filter + activation + graph pooling
- Graph filter:
 - Spatial-based
 - Spectral-based



25 YEARS ANNIVERSARY
SOICT

VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG
SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

**Thank you
for your
attentions!**



soict.hust.edu.vn/



fb.com/groups/soict

